Hypothetical Temporal Queries in Databases

Marcelo Arenas and Leopoldo Bertossi
Computer Science Department
School of Engineering
Catholic University of Chile
Casilla 306, Santiago 22, Chile.
{marenas,bertossi}@ing.puc.cl

Abstract

This paper considers the problem of posing and answering hypothetical temporal queries to databases. The queries are hypothetical in the sense that we pose a query to a virtually updated database, and the query is answered on the basis of the initial, physical database and the list of transactions that virtually update the database. The queries are temporal in the sense that they refer to possibly all the states along which the database evolves from the initial database and the final virtual state. The possibility of answering such queries relies on a specification of the dynamics of the database augmented with a specification of the dynamics of auxiliary tables that have their origin in the temporal subformulas of the query and encode the history of the database. Queries are posed in first order past temporal logic. These functionalities are implemented in SCDBR, an automated reasoner for specifications of database updates. Although we concentrate mostly on relational databases, the methodology can be applied to other kinds of databases, as deductive and first order databases.

Proceedings of the 5th KRDB Workshop
Seattle, WA, 31-May-1998

(A. Borgida, V. Chaudhri, M. Staudt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

1 Introduction

The main subject of this paper is that of modeling and doing hypothetical reasoning in databases, a subject that deserves more attention (but see [Bon90, Bon97]). We think hypothetical reasoning will become more crucial as databases are increasingly used for decision support, where "what if" questions become relevant, as the ones emerging in on-line analytical processing (OLAP) and data warehousing [Cha97]. To have computational tools that can be used to explore different courses of action without having to commit to any of them, in particular, without having to physically update the database, seems to be of major importance in many applications of information systems.

We are motivated by the problem of answering queries about different states in the evolution of a relational database, when the database is virtually updated by the execution of a sequence of primitive transactions. For example, we want to consider queries of the form "Has it always been the case that the database has satisfied a given condition C?", or "Has there been a state of the database where a certain condition C has been satisfied?", or "Has the salary of some employee decreased along the database evolution?". Although there is no explicit time in these queries we call them "temporal queries". The temporal queries we are concerned with are "hypothetical" in the sense that we have a physical database at an initial state, S_0, and a list, T, of primitive transactions $A_1, \ldots, A_n$, that virtually update the database, producing new states $S_1, \ldots, S_n$; and we want to answer a query about this sequence of states, without physically updating the whole database accordingly (and without keeping the data associated to each intermediate state). We are interested in querying this whole virtual evolution of the database.

We start from an initial database plus a specification of the dynamics of the database that describes how the tables in it evolve as primitive transactions are executed. The formalism we considered for specifying the transaction based updates of the relational database is the one shown by Reiter [Rei95], who proposed a particular kind of axioms written in the situation calculus (SC) [McC69]. In [Ber94, Ber98] the implementation and functionalities of a computational system for doing automated reasoning from and about these specifications are reported.

Here we present a solution to the problem of answering hypothetical temporal queries that relies on: (1) a formulation of a query as a query stated in first order past temporal logic (FOPTL)[1], (2) a reformulation of Chomicki's history encoding methodology for efficiently checking temporal integrity constraints [Cho95], in the context of SC specifications of database updates, and, in particular, (3) a specification in the SC of the evolution of new history encoding auxiliary relations, or views, that are generated from the query.

2 The Dynamics of a Database

The situation calculus is a family of languages of many sorted predicate logic used to represent knowledge and reason about dynamic domains that are subject to discrete changes caused by action executions. In its languages, we find domain individuals, states and actions or primitive transactions [2] at the same first order object level, then first order quantifications over all these sorts of individuals are possible. They are usually denoted by $\forall \bar{x}$, $\forall s$, $\forall a$, respectively. In every SC language we find a name for the initial state, S_0, and a function symbol, do, so that $do(a, s)$ denotes the successor state that results from the execution of action a at state s. We also find a predicate $Poss(a, s)$ with the intended meaning that action a is possible at state s. In a particular SC language we will find function names for parameterized primitive transactions, $A(\bar{x})$, and names for tables, $F(\bar{x}, s)$, that is, predicates with a single state argument. If T is a sequence of action terms $A_1, \ldots, A_n$, to be executed in that order, we abbreviate the situation $do(A_n \cdots do(A_1, S_0) \cdots)$ by $do(T, S_0)$.

The specification, Σ, of the dynamics of a database contains: (1) the *foundational axioms of the situation calculus* [Lin94], including an induction axiom on states; (2) Unique names axioms for actions and states; (3) A set, Σ_0, of SC sentences that do not mention any state term other than S_0. This is knowledge about the initial state, and state independent knowledge; (4) Axioms describing preconditions for executing (legal) primitive transactions; (5) A specification of an accessibility relation between states: $s_1 \leq s_2$ means that state s_2 can be reached from state s_1 by executing a finite sequence of legal actions; (6) And the most important ingredients, the Successor State Axioms (SSA): For every table $F(\bar{x}, s)$, an axiom of the form $\forall a, s\ Poss(a, s) \supset \forall \bar{x}[F(\bar{x}, do(a, s)) \equiv \Phi_F(\bar{x}, a, s)]$, where Φ_F is a formula to be evaluated at the execution state s. Provided there is complete knowledge at the initial state, as is usually the case in relational databases, this axiom completely determines the contents of table F at an arbitrary legal database state, i.e. reached from S_0 by a finite sequence of transactions that are possible at their execution states.

Example 1. Let us consider a database of a company with the relations $Emp(x, s)$, meaning, at state s of the database, person x is registered in the company as an employee. We also have the primitive transactions: (a) $hire(x)$, for hiring person x by the company. (b) $fire(x)$, for firing person x by the company.

The initial database, at S_0, contains the following entries:

$$Emp(john, S_0),\ Emp(ernest, S_0),\ Emp(paige, S_0).$$

In addition, the specification the following SSAs for the table Emp:

$$Poss(a, s) \supset (Emp(x, do(a, s)) \equiv a = hire(x) \vee \\ Emp(x, s) \wedge \cdot\ a \neq fire(x)),$$

and the following precondition axioms for the primitive transactions:

$$Poss(hire(x), s) \equiv \neg Emp(x, s),$$
$$Poss(fire(x), s) \equiv Emp(x, s).$$

□

Among others we find the following advantages in using the SC as a specification language: (1) It has a clear and well understood semantics; (2) Everything already done in the literature wrt applications of predicate logic to DBs can be done here, in particular, all static and extensional aspects of databases and query languages are included; (3) Dynamic aspects at the same object level can be considered, in particular, it is possible to specify how the database evolves as transactions are executed; (4) It is possible to reason in an automated manner from the specification and to extract algorithms for different computational tasks from it; (5) In particular, it is possible to reason explicitly about DB transactions and their effects; (6) In this form it is possible to extend functionalities of usual commercial DBMSs.

[1] As shown in [Are98a], all we do here can be done directly in the SC. Nevertheless, queries in temporal logic seem to be more readable, and more importantly, their syntactic structure makes the problem of detecting temporal subqueries easier.

[2] We do not make any distinction between states and situations. Primitive transactions are to be executed as a whole, they are not decomposable.

3 Temporal Queries

In the context of DB specifications like above, a temporal query is a SC sentence φ in which all the states involved, including quantified states, lie on a finite state path $S_0 \leq \cdots \leq S_n$, obtained by executing a sequence of ground actions terms $A_1, \ldots, A_n$. The query is true if and only if $\Sigma \models \varphi$.

FOPTL, first order past temporal logic, is an extension of first order logic. In its languages time is implicit, discrete and linearly ordered; formulas are evaluated on structures at a given time (found at the semantical level, but not at the object level). We find formulas of the form: (a) $\bullet\varphi$, with the intended meaning "φ was true in the previous time", and (b) φ **since** ψ, meaning "ψ became true at some time in the past and from that time on, φ has been true". Other temporal operators can be defined: (c) $\Diamond\varphi \;=_{def}\;$ *True* **since** φ, with the intended meaning "Sometime in the past φ was true"; and (d) $\Box\varphi \;=_{def}\; \neg\Diamond\neg\varphi$, with the intended meaning "Always in the past φ was true".

Every model of the situation calculus restricted to a finite path in the SC tree starting at the initial state gives rise to a sequence of first order structures (databases) that can be used to interpret FOPTL formulas. In this case, we are identifying the linear discrete time with the sequence of states. We may write $D_{S_0}, \ldots, D_{S_i}, \ldots, D_{S_n}; S_i \models \varphi$, meaning that the sentence is true at state S_i in the trajectory of databases. Since we have a past temporal logic, in this case the states $S_{i+1}, \ldots, S_n$ are not considered. As we will see below, we will always start evaluating formulas at the final state S_n reached by execution of a fixed transaction list, then, depending on the query, we will have to evaluate its temporal subformulas at previous states.

There is a sort of equivalence between FOPTL and SC for our particular kind of SC queries, which have all the explicitly and implicitly mentioned states linearly ordered and bounded above by a current state (usually a virtual one). This is because they refer to the states arising from the execution of a particular sequence of primitive transactions. Then, given a sequence of database states $D_{S_0}, \ldots, D_{S_n}$ originated by the execution of a sequence $T:\ A_1, \ldots, A_n$ of primitive transactions, for every temporal query in the SC about these states, there exists a FOPTL query to be posed at the final state S_n (wrt to the sequence of structures D_{S_0}, ..., D_{S_n}) that has the same answer. On the other side, for every FOPTL formula to be evaluated at state S_n, there is a SC temporal formula mentioning the sequence of states $S_0, \ldots, S_n$ that gives the same answer. In both cases, we say that the formulas are *answer–equivalent* in S_n.

Example 2. The query *Has Sue been working in department 13 in all states generated by sequence T at S_0?* is expressed in the SC by the formula $\forall s\ (S_0 \leq s \leq S_n \supset Emp(sue, 13, s))$, which is answer–equivalent in state S_n $(= do(T, S_0))$ to the FOPTL sentence $Emp(sue, 13) \wedge \cdot \Box Emp(sue, 13)$. $\Box$

The main reason for using FOPTL versions of queries is that it is easier from the syntactical point of view to process them with the purpose of introducing auxiliary views according to the occurrences of temporal subformulas. The same could be done directly with SC queries, but the processing of formulas would be more complicated [Are98a]. So, we will think of using temporal logic as the original query language. In any case, at the underlying reasoning level we stick to the SC formalism and logic.

4 The Dynamics of History Encoding

In [Cho95] the problem of checking temporal constraints stated in FOPTL was considered. These are constraints that talk about, and relate, different states of the database. There we find a sequence of transactions that are physically executed, and in order to minimize the cost of checking, one progressively updates new defined relations, or auxiliary views, r_α's, that correspond to the temporal subformulas, α's, in the constraint. These views encode part of the database evolution up to a current database state. They are defined and updated in such a way that they store the historical information that is relevant to give an answer to the query about the satisfaction of the integrity constraint once the final (current) state is reached. Then a new, non temporal, but local and static query can be posed at the final state. Here we combine our reconstruction of history encoding in the context of specifications of database updates with the possibility, opened by those specifications, of reasoning about the database evolution, without having to physically update the database. Then, we are in position of doing hypothetical temporal reasoning. We can say that while in [Cho95] the query is answered by positioning at the final physical state of the database, we query a single future, virtual state from the initial, physical state. In this way, virtual updates make possible to apply history encoding to any temporal query, whereas with physical updates it can be applied to fixed, predetermined temporal queries only.

Our starting point consists of a SC specification Σ and a FOPTL sentence φ to be evaluated at the final state $S = S_n = do(T, S_0)$, that results from the virtual update of the DB. So, φ implicitly refers to the states between S_0 and S. We are able to obtain a new SC specification Σ_H that extends Σ, and a SC sentence $\varphi^h(S)$, such that the answer to the original query φ coincides with the answer obtained from the evaluation of $\varphi^h(S)$ wrt Σ_H. (Here the super and sub indices ^{h} and $_H$ stand for "history".) The resulting SC sentence refers explicitly to the state S only, and Σ_H contains a specification of the dynamics of new, history encoding, auxiliary relations of the kind introduced in [Cho95]. This is done by means of derived successor state axioms for the auxiliary views.

Notice that φ^h is instantiated at the final state $do(T, S_0)$, and this is the only state mentioned in the formula. Then, we have transformed the problem of answering a temporal query wrt to a virtually updated database into the *temporal projection problem* of artificial intelligence [Han86], that is, into the problem of querying a single future state obtained by the execution of an actions sequence.

Now we will sketch how to construct φ^h, first, and Σ_H, next. The construction of φ^h is done by induction on φ. We will introduce new SC tables for each temporal subformula of the FOPTL formula $\varphi(\bar{x})$.
1. If φ is of the form $F(\bar{t})$, then $\varphi^h =_{def} F(\bar{t}, S)$.
2. If φ is $\neg\psi$, then $\varphi^h =_{def} \neg(\psi^h)$.
3. $(\varphi * \psi)^h =_{def} (\varphi^h * \psi^h)$, where $*$ is any of the usual binary propositional connectives.
4. $(Q\varphi)^h =_{def} Q(\varphi^h)$, where Q is any of the usual first order quantifiers.
5. If φ is $\bullet\psi(\bar{x})$, with ψ non temporal, then $\varphi^h =_{def} R_\varphi(\bar{x}, S)$, where R_φ is a new table name.
6. If $\varphi(\bar{x})$ is $\psi(\bar{x})$ **since** $\theta(\bar{x})$, where $\psi(\bar{x})$ and $\theta(\bar{x})$ are non temporal formulas, then $\varphi^h =_{def} R_\varphi(\bar{x}, S)$, where R_φ is a new table name.

By bottom–up transformation of a FOPTL formula φ, we can always obtain such a φ^h. Notice that formula φ^h is a SC formula containing a free state variable s which is its only state term.

Now we specify the dynamics of the new, auxiliary tables by means of SSAs. For this, we have to to consider the last two cases in the previous inductive definition:
(a) Let $\alpha(\bar{x})$ be of the form $\bullet\psi(\bar{x})$. This formula is true at a given state iff $\psi(\bar{x})$ is true at the previous state. Then, the corresponding new table, $R_\alpha(\bar{x}, s)$, has the SSA

$$\forall(a,s)Poss(a,s) \supset \forall\bar{x}\ (R_\alpha(\bar{x}, do(a,s)) \equiv \psi^h(\bar{x}, s)).$$

At the initial state $\alpha(\bar{x})$ is false for each $\bar{x}$, because there is no state previous to S_0. So, we define $\forall\bar{x}\ \neg R_\alpha(\bar{x}, S_0)$.
(b) Let $\alpha(\bar{x})$ be of the form $\psi(\bar{x})$ **since** $\theta(\bar{x})$. It is not difficult to see that for its associated table, $R_\alpha(\bar{x}, s)$, it holds

$$\forall(a,s)Poss(a,s) \supset \forall\bar{x}(R_\alpha(\bar{x}, do(a,s)) \equiv \\ \psi^h(\bar{x}, do(a,s)) \wedge (R_\alpha(\bar{x}, s) \vee \theta^h(\bar{x}, s)).$$

This is not a SSA, because there is a $do(a,s)$ term in ψ^h on the RHS. Nevertheless, we can get rid of it applying Reiter's regression operator $\mathcal{R}$, that takes a formula, instantiated at a successor state of the form $do(A, s)$, into a formula instantiated at the previous state, s. For doing this, $\mathcal{R}$ uses the SSAs of the tables appearing in $\psi^h(\bar{x}, do(a,s))$. This operator preserves logical equivalence [Rei95].

In consequence, we obtain:

$$\forall(a,s)Poss(a,s) \supset \forall\bar{x}(R_\alpha(\bar{x}, do(a,s)) \equiv \\ \mathcal{R}[\psi^h(\bar{x}, do(a,s))] \wedge (R_\alpha(\bar{x}, s) \vee \theta^h(\bar{x}, s))).$$

As before, we also define $\forall\bar{x}\ \neg R_\alpha(\bar{x}, S_0)$.

It is possible to prove that $D_0, \ldots, D_n; S_n \models \varphi$ is equivalent to $\Sigma_H \models \varphi^h(do(T, S_0))$, where T is a list of primitive transactions that virtually update the database D_0 producing virtual databases $D_1, \ldots, D_n$, being S_n the last reached state. We emphasize that φ^h is evaluated at this only state.

We have added to our automated reasoner, SCDBR, the functionality of of generating the new SC formula and specification, including the application of the regression operator [Ber98]. The reasoner is implemented in PROLOG.

Example 3. Let us consider the FOPTL formula

$$\Diamond p(v_1)\ \textbf{since}\ \Box q(v_1). \tag{1}$$

From it we want to generate a SC formula $\psi(v_1, s)$, such that (1) is answer–equivalent to $\psi(v_1, s)$ in every legal state s. For this we will use some of the procedures of SCDBR.

```
| ?-  i_p(diamond p(v1)  since  box q(v1),F),
tl_initial(F,s,I1, [F2,F3,F4],_, [F5,F6,F7]),
p_i([F2,F3,F4,F5,F6,F7],[I2,I3,I4,I5,I6,I7]).

I1= r_0(v1,s),
I2= forall(v1):(neg r_0(v1,s0)),
I3= forall(v1):(neg r_0_sl(v1,s0)),
I4= forall(v1):(neg r_0_sr(v1,s0)),
I5= forall(a):( poss(a,s) => (r_0(v1,do(a,s))
    <=>(r_0_sl(v1,s) v p(v1,s)) &(r_0(v1,s) v
    neg r_0_sr(v1,s)))),
I6= forall(a):(poss(a,s)=>(r_0_sl(v1,do(a,s))
    <=> r_0_sl(v1,s) v p(v1,s))),
I7= forall(a):(poss(a,s)=>(r_0_sr(v1,do(a,s))
    <=> r_0_sr(v1,s) v neg q(v1,s)))
```

Firstly, we introduce (1) by means of the predicate `i_p`, which translates the formula into, `F`, its internal representation in the system. Secondly, the predicate `tl_initial` is applied on `F`, obtaining: (a) a formula answer-equivalent to F, which is stored in I1; (b) the list of the initial state formulas for the generated auxiliary tables, which are stored in F2, F3 and F4; (c) the list of the successor state axioms for the generated auxiliary tables, which are stored in F5, F6 and F7. Finally, we the predicate p_i is applied for translating (b) and (c), written in the internal representation, into I2, I3, I4, I5, I6 and I7, written in infix notation.

From the SCDBR's invocation, we can see that the formula that is answer–equivalent to (1) is

$$r\_0(v_1, s).$$

For producing this formulas, the system generated three auxiliary tables r_0, r_0_sl and r_0_sr, with the following initial state definitions[3]:

$$\begin{aligned}&\forall v_1(\neg r\_0(v_1, S_0)),\\ &\forall v_1(\neg r\_0\_sl(v_1, S_0)),\\ &\forall v_1(\neg r\_0\_sr(v_1, S_0));\end{aligned}$$

with the following SSAs axioms:

$$\begin{aligned}&Poss(a, s) \supset (r\_0(v_1, do(a, s)) \equiv (r\_0\_sl(v_1, s) \vee \\ &\qquad p(v_1, s)) \wedge\cdot (r\_0(v_1, s) \vee \neg r\_0\_sr(v_1, s))),\\ &Poss(a, s) \supset (r\_0\_sl(v_1, do(a, s)) \equiv r\_0\_sl(v_1, s) \vee \\ &\qquad p(v_1, s)),\\ &Poss(a, s) \supset (r\_0\_sr(v_1, do(a, s)) \equiv r\_0\_sr(v_1, s) \vee \\ &\qquad \neg q(v_1, s)).\end{aligned}$$

□

5 Answering Queries

To solve the problem of answering the resulting, new query wrt the extended specification, it is possible to apply some existing techniques for Reiter like specifications, e.g. Reiter's query regression [Rei95], minimal rolling forward of the database based on information that is relevant to the query [Ber98, Siu96], or even full progression of the database [Lin97]. All these methodologies are supported by SCDBR. For example, if we run the regression procedure in SCDBR to solve the temporal projection problem, we will eventually obtain a query to be posed to the initial database. SCDBR can answer this query by calling a conventional DBMS (actually, ORACLE) or a PROLOG program or a theorem prover (OTTER), depending on the kind of initial, physical database available.

Example 4. Let us consider the specification shown in example 1. We want to know the list of the employees who have always been working in the company, in all states generated by the execution of the sequence of actions $T = [hire(sue), fire(john)]$ from the initial situation. Thus, we are asking

$$\forall s(S_0 \leq s \leq do(T, S_0) \supset Emp(x, s)). \quad (2)$$

This formula is answer-equivalent in $do(T, S_0)$ to the FOPTL formula

$$Emp(x) \wedge\cdot \Box Emp(x). \quad (3)$$

We can answer this query by means of the following invocation of SCDBR

```
| ?- i_p(emp(x) & box emp(x),F),tl_initial(F,
do(fire(john), do(hire(sue), s0)), F1,[F2],I,
[F3]), p_i(F1, F6), p_i(F2, F7), p_i(F3, F8),
reg_n(F1, 2, F4),prune_una(F4,F5),p_i(F5,F9),
prolog_initial(F5,I).

Transforming Query into Prolog Goal...Done.

[[x,ernest]]
[[x,paige]]

F6 = emp(x,do(fire(john),do(hire(sue),s0))) &
     neg r_0_r(x, do(fire(john),do(hire(sue),
     s0))),
F7 = forall(x):(neg r_0_r(x,s0)),
F8 = forall(a):(poss(a,s)=> (r_0_r(x,do(a,s))
     <=> r_0_r(x,s)v neg emp(x,s))),
F9 = ((sue eq x v emp(x,s0)) & neg john eq x)
     & neg( (r_0_r(x, s0) v neg emp(x, s0)) v
     neg (sue eq x v emp(x,s0)))
```

As in example 3, we use the predicate tl_initial for generating a formula that is answer-equivalent to (3). More precisely, the system generates F6:

$$\begin{aligned}&Emp(x, do(fire(john), do(hire(sue), S_0))) \wedge\cdot\\ &\neg r\_0\_r(x, do(fire(john), do(hire(sue), S_0))), \quad (4)\end{aligned}$$

which includes the auxiliary table r_0_r. For this table, the system also generates the following formula describing its initial state

$$\forall x(\neg r\_0\_r(x, S_0)), \quad (5)$$

and the following SSA

$$\begin{aligned}&Poss(a, s) \supset (r\_0\_r(x, do(a, s)) \equiv\\ &\qquad (r\_0\_r(x, s) \vee \neg Emp(x, s))). \quad (6)\end{aligned}$$

In consequence, for answering (2), (4) needs to be answered wrt the company database plus the information in (5) and (6).

[3] SCDBR can automatically produce Latex formulas as output.

As we see in the invocation, the procedure reg_n is used for applying twice the regression operator on (4). After that, by means of the predicate prune_una, the resulting formula is simplified on the basis of the unique name axioms for actions. So, the following formula F9 is generated to be posed against the initial database state

$$((sue = x \vee Emp(x, S_0)) \wedge\cdot \neg john = x) \wedge\cdot \neg((r\_0\_r(x, S_0) \vee \neg Emp(x, S_0)) \vee \neg(sue = x \vee Emp(x, S_0))). \quad (7)$$

If the (initial) database of the company is a PROLOG database, we answer to (7) by means of the procedure prolog_initial, which uses PROLOG as query language. In this way, we obtain the following tuples as answer to the query:

$$x = ernest \vee x = paige.$$

□

As described in [Ber98], SCDBR can be interfaced with a RDBMS.

Example 5. We have an ORACLE database with information about the employees in the company at the initial state. We execute the list of transactions $[fire(john), fire(ernest)]$. Now we want to know all the employees who have worked for the company from the initial state on:

$$\exists s(S_0 \leq s \leq do(T, S_0) \wedge\cdot Emp(x, s)).$$

In FOPTL this query is expressed by

$$Emp(x) \vee \Diamond Emp(x),$$

and has to be posed at the state $do(T, S_0)$.

We can ask SCDBR to generate a new query in SQL to be posed against the initial relational database:

```
| ?- i_p(emp(x) & box emp(x),F),tl_initial(F,
do(fire(john), do(hire(sue), s0)), F1,[F2],I,
[F3]), p_i(F1, F6), p_i(F2, F7), p_i(F3, F8),
reg_n(F1, 2, F4),prune_una(F4,F5),p_i(F5,F9),
ora_sql(F5).

[[paige],[john],[ernest]]
```

Here F5 stands for a first order query that the predicate ora_sql transforms, via a lower level predicate, into a SQL query to be posed to the ORACLE database. After that, ora_sql prints the answer. In this case, we obtain:

$$x = paige \vee x = john \vee x = ernest$$

□

6 Another Application: Transforming Dynamic ICs

In [Cho95] it is shown how to check dynamic integrity constraints (ICs) statically. This was the main motivation for the introduction of history encoding.

In the context of specifications of database updates, as in [Rei95], we expect integrity constraints to be logical consequences of the specification. From this point of view, methodologies for automatically proving static ICs from the specification have been developed [Ber96]. It turns out that with our treatment of history encoding in specifications of database updates in terms of the dynamics of history encoding relations, we can transform dynamic ICs into static ICs, and keep logical consequence from the specification. In particular, dynamic ICs can be proved as static ICs.

Assume that ψ is dynamic IC, that is it is a sentence one expects to be true at every particular legal state s of the database. In ψ there may be quantifications on several states, so it does not talk about the current state only. Nevertheless, while dynamic ICs talk about several states of the DB, they have to be satisfied locally. We can use this idea in the transformation mechanism.

From ψ we generate a new constraint

$$\forall s(S_0 \leq s \supset \psi'(s)),$$

where $\psi'(s)$ is obtained from ψ by relativising all the quantifications over states to $\leq s$, e.g. $\exists s'$ becomes $\exists s' \leq s$. If $\psi'(s)$ has an answer–equivalent FOPTL sentence φ at s, then we generate the static IC

$$\forall s(S_0 \leq s \supset \varphi^h(s)),$$

and a new specification Σ_H, where φ^h and Σ_H are constructed as in section 4. It is possible to prove that $\Sigma \models \psi$ iff $\Sigma_H \models \forall s \; (S_0 \leq s \supset \varphi^h(s))$.

Example 6. Consider the specification Σ of the employees DB with a salary table $Sal(x, p, s)$. We want the usual dynamic IC to follow from Σ:

$$\psi: \quad \forall(s', s'')(S_0 \leq s' < s'' \supset \forall(x, p', p'') \; Sal(x, p', s') \wedge\cdot Sal(x, p'', s'')) \supset p' \leq p''). \quad (8)$$

This IC should hold in every legal current state s of the DB. We replace ψ by a new formula $\psi'(s)$:

$$\forall(s', s'')(S_0 \leq s' < s'' \leq s \supset \forall(x, p', p'') \; ((Sal(x, p', s') \wedge\cdot Sal(x, p'', s'')) \supset p' \leq p'')).$$

Formula (8) holds in every legal state if and only $\forall s \; (S_0 \leq s \supset \psi'(s))$ follows from Σ. Now, transform $\psi'(s)$ into a new formula $\varphi^h(s)$ that can be statically evaluated against the new specification Σ_H generated from Σ.

Formula $\psi'(s)$ is answer–equivalent at s to the following FOPTL sentence

$$\forall(x,p',p'')((\Diamond Sal(x,p') \wedge\cdot\ Sal(x,p'')) \supset p' \leq p'') \wedge\cdot \\ \Box\forall(x,p',p'')((\Diamond Sal(x,p') \wedge\cdot\ Sal(x,p'')) \supset p' \leq p'').$$

Applying the methodology of section 4, we obtain the static IC

$$\forall s(S_0 \leq s \supset [\forall(x,p',p'') \\ ((R_\alpha(x,p',s) \wedge\cdot\ Sal(x,p'',s)) \supset p' \leq p'') \wedge\cdot R_\beta(s)])$$

and the new specification contains the following axioms for the new tables R_α and R_β (a propositional state dependent formula):

(a) $\forall(x,p')\neg R_\alpha(x,p',S_0)$,

(b) $\forall(a,s)Poss(a,s) \supset \forall(x,p')\ (R_\alpha(x,p',do(a,s)) \equiv R_\alpha(x,p',s) \vee Sal(x,p',s))$,

(c) $R_\beta(S_0)$,

(d) $\forall(a,s)Poss(a,s) \supset R_\beta(do(a,s)) \equiv R_\beta(s) \wedge\cdot \forall(x,p',p'')((R_\alpha(x,p',s) \wedge\cdot\ Sal(x,p'',s)) \supset p' \leq p'')$.

□

7 Conclusions and Further Work

Among the contributions in this paper we find the following: (1) An extension of Chomicki's methodology for checking temporal integrity constraints to the situation where a specification of the evolution of the database is available; and so (2) The possibility of doing hypothetical reasoning along a virtual evolution of the database ([Cho95] concentrates on physical updates of the database), and this with user defined primitive transactions; (3) A general solution to the problem of answering temporal queries in the context of Reiter's specifications of database updates, and this solution works both in a progressive as in a regressive way; (4) An implementation of the query mechanisms in the context of an automated reasoner for specifications of database updates.

It turns out that the methodology we developed for answering queries can be used to give solutions to other reasoning problems. We are in position to transform dynamic integrity constraints into static integrity constraints. The original dynamic constraint is a logical consequence of the original specification of the database dynamics if and only if the new, generated static integrity constraint is a logical consequence of the original specification extended with an specification of the dynamics of auxiliary history encoding relations. In particular, any available methodology for handling static integrity constraints can be adapted for the dynamic case. For example, we can take advantage of results on automated proving of static integrity constraints [Ber96, She89] when dealing with the dynamic case.

As described in [Are98a], another problem we are in position to solve consists in transforming preconditions for action executions that depend on the history of the database into preconditions that depend on the local, execution state.

It is matter of our current research an extension of the methodology of this paper that includes metric time as in [Cho95]. This is done by considering actions parameterized with explicit time, as in [Rei96]. We are also working on complexity analysis of the methodologies used to solve the resulting temporal projection problem. Another issue has to do with coping with the problem of safeness of the resulting, transformed queries. We are also interested in applying derived SSAs for views [Are98b] as an alternative to explicit regression with the intention of generating simpler queries to be posed to the initial database.

Acknowledgments

This research has been partially supported by FONDECYT Grants (#1971304 and #1980945), DIPUC, DAAD, GK "Verteilte Informationssysteme", TUBerlin (CIS). We are grateful to Mauricio Strello for helping us run the examples.

References

[Are98a] M. Arenas and L. Bertossi. Hypothetical temporal reasoning with history encoding. To appear in *Proc. of "ESSLLI-98 Workshop on Reasoning about Actions: Foundations and Applications"*, Saarbruecken, August 17 - 21, 1998.

[Are98b] M. Arenas and L. Bertossi. The dynamics of database views. To appear in *Transactions and Change in Logic Databases*, H. Decker, B. Freitag, M. Kifer, A. Voronkov (eds.), Springer, 1998.

[Ber94] L. Bertossi and C. Ferretti. SCDBR: A reasoner for specifications in the situation calculus of database updates. In *Temporal Logic. Proceedings First International Conference, ICTL '94, Bonn, Germany, July 1994, LNAI 827*, pp. 543–545. Springer, 1994.

[Ber96] L. Bertossi, J. Pinto, P. Saez, D. Kapur, and M. Subramaniam. Automating proofs of integrity constraints in the situation calculus. In *Foundations of Intelligent Systems. Proc. Ninth International Symposium on Methodologies for Intelligent Systems (ISMIS'96), Zakopane, Poland*, pp. 212–222. Springer, LNAI 1079, 1996.

[Ber98] L. Bertossi, M. Arenas and C. Ferretti. SCDBR: An automated reasoner for specifications of database updates. *Journal of Intelligent Information Systems*, 10(3), 1998.

[Bon90] A. Bonner. Hypothetical datalog: Complexity and expressivity. *Theoretical Computer Science*, 76(1):3–51, 1990.

[Bon97] A. Bonner and M. Kifer. Transaction Logic Programming. In *Logics for Databases and Information Systems*, J. Chomicki and G. Saake (eds.). Kluwer Academic Publishers, 1998.

[Cha97] S. Chaudhuri and U. Dayal. An overview of datawarehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, March 1997.

[Cho95] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems*, 20(2):149-186, June 1995.

[Han86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. *Proc. National Conference on Artificial Intelligence*, pp. 328-333, 1986.

[Lin94] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Action and Processes.

[Lin97] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1–2): 131–167, 1995.

[McC69] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie (eds.), *Machine Intelligence*, volume 4, pp. 463–502, Edinburgh, Scotland, 1969. Edinburgh University Press.

[Rei95] R. Reiter. On specifying database updates. *Journal of Logic Programming*, 25(1):53–91, 1995.

[Rei96] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, Morgan Kaufmann, 1996.

[She89] T. Sheard and D. Stemple. Automatic verification of database transaction safety. *ACM Transactions on Database Systems*, 14(3):322–368, 1989.

[Siu96] B. Siu and L. Bertossi. Answering historical queries in databases (extended abstract). In *Proc. XVI International Conference of the Chilean Computer Science Society (SCCC'96)*, M. V. Zelkowitz and P. Straub (eds.), Valdivia, Nov. 13–15, 1996.